International Journal of Remote Sensing, 2013
Vol. 34, No. 23, 8408–8424, http://dx.doi.org/10.1080/01431161.2013.838710

Object class segmentation of massive 3D point clouds of urban areas using point cloud topology

Rico Richter*, Markus Behrens, and Jürgen Döllner

Hasso-Plattner-Institut, Potsdam, Germany

(*Received 7 February 2013; accepted 17 August 2013*)

A large number of remote-sensing techniques and image-based photogrammetric approaches allow an efficient generation of massive 3D point clouds of our physical environment. The efficient processing, analysis, exploration, and visualization of massive 3D point clouds constitute challenging tasks for applications, systems, and workflows in disciplines such as urban planning, environmental monitoring, disaster management, and homeland security. We present an approach to segment massive 3D point clouds according to object classes of virtual urban environments including terrain, building, vegetation, water, and infrastructure. The classification relies on analysing the point cloud topology; it does not require per-point attributes or representative training data. The approach is based on an iterative multi-pass processing scheme, where each pass focuses on different topological features and considers already detected object classes from previous passes. To cope with the massive amount of data, out-of-core spatial data structures and graphics processing unit (GPU)-accelerated algorithms are utilized. Classification results are discussed based on a massive 3D point cloud with almost 5 billion points of a city. The results indicate that object-class-enriched 3D point clouds can substantially improve analysis algorithms and applications as well as enhance visualization techniques.

1. Introduction

Remote-sensing technologies, such as lidar (Wehr and Lohr 1999), capture our physical environment, e.g. cities and landscapes. In addition, image-based photogrammetric approaches (Leberl et al. 2010) are rapidly evolving (Snavely et al. 2008) (e.g. Autodesk's 123D or Microsoft's Photosynth). The resulting 3D point clouds describe the surface of a captured region in a discrete way by independent unstructured points.

In many disciplines, massive 3D point clouds represent an essential category of data for applications, systems, and workflows. The efficient processing is a challenging task for applications in urban planning, environmental monitoring, disaster management, homeland security, and navigation systems. 3D point clouds represent also basic data used to derive 3D building (Zhou and Neumann 2008; Meixner, Leberl, and Brédif 2011) and terrain models (Meng, Currit, and Zhao 2010). In general, 3D point clouds provide only geometric data (i.e. x, y, z coordinates).

Object class segmentation provides a way to split a massive 3D point cloud into disjoint subsets belonging to different object classes. For urban areas, these classes typically

*Corresponding author. Email: rico.richter@hpi.uni-potsdam.de

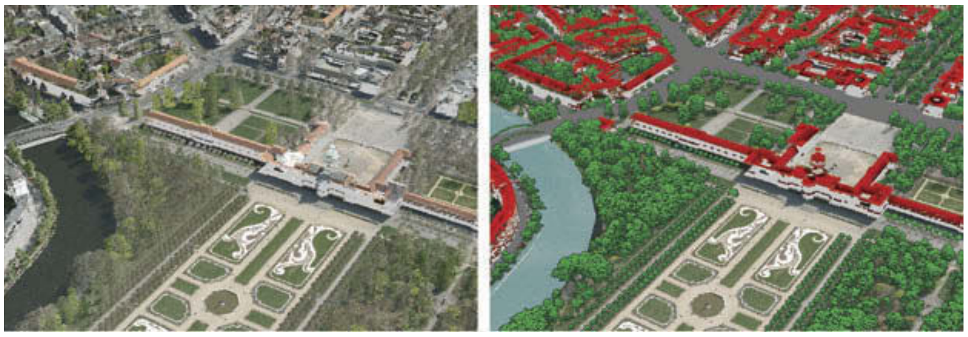


Figure 1. Illustration of a 3D point cloud before and after object class segmentation. The resulting object classes are illustrated with their respective colours: buildings, red; vegetation, green; water, blue; infrastructure, grey; terrain, aeriel image colours. (*a*) 3D point cloud with colour information. (*b*) 3D point cloud with object class information.

include categories such as terrain, building, vegetation, water, and infrastructure (Figure 1). The process of enhancing 3D points by this kind of semantics is called *point cloud classification* (Lodha, Fitzpatrick, and Helmbold 2007; Carlberg et al. 2009; Samadzadegan, Bigdeli, and Ramzi 2010). The classification can be used to optimize and enhance algorithms operating on massive 3D point clouds. Most importantly, a classified 3D point cloud can significantly reduce the required amount of data and improve the performance and accuracy of algorithms. For example, building reconstruction, infrastructure monitoring, or flood simulations can operate on a subset of the overall 3D point cloud belonging to object classes that are relevant for an application.

Our approach classifies massive 3D point clouds according to principal object classes found in urban areas. It is based on analysing the *3D point cloud topology*, i.e. geometric relationships between points and segments, such as connectivity, local flatness, smoothness, and orientation. We use a combination of feature- and segment-based algorithms. At first, ground points are detected. Second, large segments with a characteristic topology for vegetation and building structures are determined. Third, all unclassified segments are analysed within multiple passes taking already classified points into account. Finally, a classification of ground points is performed using geospatial data with semantics information (e.g. thematic maps). This determines specific object classes for ground points, such as terrain, water, and, infrastructure, that cannot be derived from the 3D point cloud topology in general.

Existing segmentation approaches frequently require additional per-point attributes (e.g. pulse or intensity information) or operate on manually classified training data sets as for machine learning approaches (Jiang, Zhang, and Ming 2008; Samadzadegan, Bigdeli, and Ramzi 2010; Lodha, Fitzpatrick, and Helmbold 2007). The presented approach does not require per-point attributes or training data to classify the data into ground, vegetation and building points. In addition, it can be applied to 3D point clouds from different remote-sensing technologies.

Due to increasing precision, decreasing costs, and higher frequency of capturing, applications and systems are faced with massive sets of 3D point clouds. Therefore, out-of-core algorithms are required to process data that do not fit into main memory or graphics processing unit (GPU) memory. Our previous work showed that spatial data structures and GPU-accelerated processing of time-consuming processes can be used to compare multiple 3D point clouds from at least two different timestamps (Richter, Kyprianidis, and Döllner 2012). By contrast, this article presents concepts and techniques for the classification of 3D point clouds.

We evaluated our processing pipeline with a real world data set with 5 billion points of the urban region of a city and compare the results with ground truth data. The accuracy evaluation was done for subsets of the 3D point cloud with different characteristics and shows a precision of at least 93.4% for vegetation, 95.2% for building, and 97.9% for ground point detection.

Object-class-enriched 3D point clouds can improve the visualization by applying different 3D rendering techniques to different object classes or by providing different interaction techniques for them. The results are presented by utilizing an out-of-core rendering system for massive 3D point clouds.

2. Related work

One category of related approaches is based on additional attributes per point such as colour (Charaniya, Manduchi, and Lodha 2004), intensity (Charaniya, Manduchi, and Lodha 2004; Lodha, Fitzpatrick, and Helmbold 2007; Samadzadegan, Bigdeli, and Ramzi 2010), scan line (Sithole and Vosselman 2005; Douillard et al. 2011), amplitude (Alexander et al. 2011), and pulse information (Charaniya, Manduchi, and Lodha 2004; Clode and Rottensteiner 2005; Samadzadegan, Bigdeli, and Ramzi 2010), which are specific for the used scanning technology and device. Other classification approaches use per-point features, segment features, or machine learning algorithms (Samadzadegan, Bigdeli, and Ramzi 2010; Lodha, Fitzpatrick, and Helmbold 2007), which require training data sets that have to be prepared manually. Our approach, however, does not require any additional information per point or manually classified training data sets.

One main purpose of processing aerial 3D point clouds is to construct digital terrain models (DTMs) (Raber et al. 2002; Shao and Chen 2008; Meng, Currit, and Zhao 2010) and 3D building models (Meixner, Leberl, and Brédif 2011; Zhou and Neumann 2008). Sithole and Vosselman (2004) performed an experimental comparison of several ground filter algorithms with different 3D point clouds from airborne laser scanning. Meng, Currit, and Zhao (2010) give a detailed overview and evaluation of ground filtering algorithms to derive digital elevation models (DEMs). Our classification is based on the approach of Rabbani, van den Heuvel, and Vosselman (2006) to derive segments in a 3D point cloud, which are analysed based on their topological structure to identify ground surface segments.

Building reconstruction algorithms require roof points as input data to derive building models. These algorithms typically require only points that belong to planar roof areas (Zhou and Neumann 2008; Meixner, Leberl, and Brédif 2011). Therefore, an exact and complete detection of all points belonging to facade structures and small roof elements, e.g. smoke stacks, roof dormers, and other roof constructions, is not necessary (Meixner, Leberl, and Brédif 2011). In contrast, the presented approach is designed to detect all building elements that can be recognized in the 3D point cloud.

Well-proved vegetation classification approaches use first and last pulse information (Clode and Rottensteiner 2005; Jiang, Zhang, and Ming 2008; Charaniya, Manduchi, and Lodha 2004), reflection intensity (Charaniya, Manduchi, and Lodha 2004), or colour attributes (Charaniya, Manduchi, and Lodha 2004) of each point. These attributes are not always available, e.g. for 3D point clouds resulting from image-based approaches. The presented approach is applicable for 3D point clouds without specific per-point attributes because it operates only on the point cloud topology.

Golovinskiy, Kim, and Funkhower (2009) use data from airborne and mobile scans to identify city furniture (e.g. newspaper boxes, traffic lights, cars) in urban environments.

Other approaches detect power lines (Clode and Rottensteiner 2005) using per-point return information and road networks (da Silva, Centeno, and Henriques 2011) using digital images.

Machine learning and support vector machines (SVMs) require a representative and manual classified 3D point cloud as training data (Jiang, Zhang, and Ming 2008; Samadzadegan, Bigdeli, and Ramzi 2010; Lodha, Fitzpatrick, and Helmbold 2007). Shapovalov, Velizhev, and Barinova (2010) and Shapovalov and Velizhev (2011) use non-associative markov networks to classify airborne and terrestrial laser scans. Lodha, Fitzpatrick, and Helmbold (2007) use per-point features (height variation, normal variation, return intensity) to group points with similar characteristics. Alexander et al. (2011) use a decision tree approach that can dynamically handle a large number of attributes. The manual preparation of training data is time-consuming and has to be accomplished for 3D point clouds that differ in their characteristics, e.g. point density and point distribution. In contrast to our approach, no specific domain knowledge is used, e.g. arrangement of ground, vegetation, and building segments.

Carlberg et al. (2009) introduce a multi-category classification system for airborne lidar data with similar objectives compared to our system. Multiple classifiers perform a region growing algorithm and a segment-wise classification. Planar and scattered segments are identified using manual classified training data. This approach results in a large number of unclassified points, especially for segments resulting from building facades and small scattered areas. We reduce the amount of unclassified points by performing an iterative multi-pass analysis.

3. Object class segmentation

A schematic overview of the object class segmentation pipeline is illustrated in Figure 2. *Preprocessing* removes outliers and duplicates and computes per-point normals.

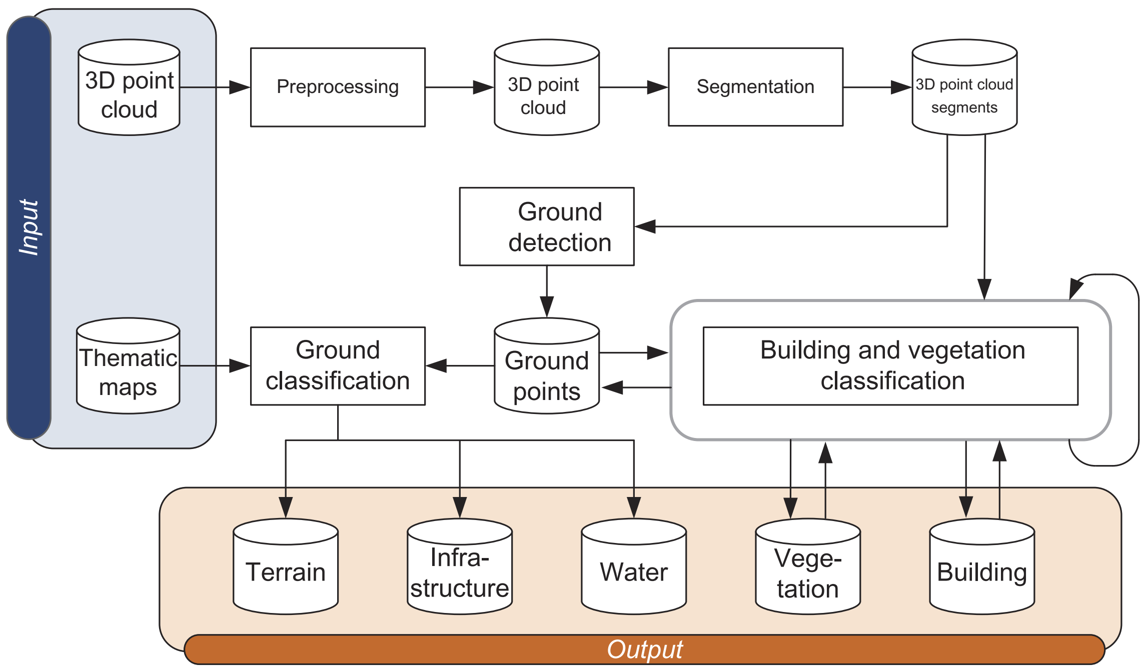


Figure 2. System overview showing the data workflow and components of the object class segmentation pipeline.

Segmentation groups points with similar characteristics in a local proximity. These segments are used for *ground detection*. *Vegetation and building detection* considers different aspects of the 3D point cloud topology and classification results of previous passes. Finally, the *ground classification* splits ground points into object classes based on thematic maps. The original 3D point cloud is subdivided into five disjoint partitions representing the object class categories building, vegetation, terrain, infrastructure, and water.

3.1. Preprocessing and segmentation

The preparation of the input data has two major aims. First, it ensures a defined quality of the 3D point cloud. For this purpose, outliers and duplicates in the data are removed. Second, new per-point attributes are added (e.g. normal and segment information) that are needed for the further processing. This requires additional temporary memory resources during processing that can be deallocated after finishing the classification. Relevant parameters for preprocessing and segmentation can be derived from point distribution and density. In the following, all mentioned parameters are related to a data set with a point density of 5–10 points/m^2 (see Section 5).

3.1.1. Filtering and normal vector computation

Outlier detection determines isolated points that often result from measurement errors. Therefore, the number of neighbouring points within a local point proximity (e.g. 2 m) is determined. If no other point can be found, the point is defined as an outlier and not considered in further processing. Duplicates are identified by comparing the point position with points in a small proximity (e.g. 0.01 m). An essential point attribute for the object class segmentation is the normal vector of each point, which approximates the surface of the local point proximity. It is computed using the covariance matrix of the nearest neighbours (e.g. 10 points) and the corresponding eigenvectors and eigenvalues (Hoppe et al. 1992).

3.1.2. Segmentation

Segmentation is a process that partitions the 3D point cloud into disjoint subsets with similar characteristics. Common segmentation approaches perform a region growing to group spatially connected points within a defined proximity (e.g. 0.5 m). The aim of the presented segmentation is to group only points belonging to the same object class (e.g. ground, building, vegetation) to enable a segment-based classification. However, points belonging to different object classes can be located next to each other (e.g. trees close to building roofs or ground surfaces close to building facades). For that reason, an additional segmentation criterion is required that considers surface structure properties. Rabbani, van den Heuvel, and Vosselman (2006) introduced a segmentation approach that considers surface smoothness in addition to the local connectivity of points. *Surface smoothness* and *residual value* parameters are used by a region growing algorithm. The normal variation between points (surface smoothness) is used to stop the segmentation at edges and offsets in the 3D point cloud. This parameter is difficult to determine for the entire 3D point because the segmentation should work for smooth regions (e.g. planar roofs) but also be tolerant to curved areas (e.g. bended building structures). For that reason, a second parameter is used that indicates areas of height curvature in the 3D point cloud (residual value). This allows regarding

Table 1. Segmentation parameters for 3D point clouds with different characteristics.

	Type	Points/m^2	Smoothness	Residual
DS1	Lidar	5–10	0.5	0.05
DS2	Lidar	10–15	0.5	0.01
DS3	2.5D Grid	100	0.5	0.005

surface smoothness and locally adjusting normal variation based on the 3D point cloud structure. Suitable segmentation thresholds for the presented data sets are 0.5 for surface smoothness and 0.05 as the residual value. An increased residual value results in larger segments that could contain multiple object class points. A decreased residual value increases the size of small segments. The residual value strongly depends on the point density. Values for 3D point clouds with different characteristics are listed in Table 1.

3.2. Ground detection

The ground detection algorithm operates on segments that result from the segmentation pass. Ground points typically form large-area segments due to smooth surface characteristics and connectivity of ground areas. In contrast, building and tree-covered areas generate smaller segments (Meng, Currit, and Zhao 2010). For that reason, large-area segments (e.g. >50 km^2) are assumed as ground segments. These segments include the majority of ground points. Special cases occur for small ground regions surrounded by buildings (e.g. backyards) or located in dense vegetated areas that are not connected with large-area ground segments. To identify those segments, all non-classified segments are analysed concerning the relative position to already classified ground points. For each point in a segment, the nearest ground points are identified. If the height difference to these ground points is below a defined threshold (e.g. 0.5 m), the point is tagged as a possible ground point. The whole segment is identified as a ground segment if the segment contains a significant number of possible ground points (e.g. 80%).

The ground detection pass outputs a 3D point cloud that represents the ground surface with a 2.5D characteristic, i.e. there are no overlapping structures. The ground surface can be subdivided into object classes such as terrain, water, and infrastructure (e.g. road networks and railway). A detailed differentiation of these ground points can rarely be derived from the 3D point cloud topology due to the smooth connection of ground surface areas. For that reason, a more specific classification of ground points can be performed by utilizing additional geospatial data, e.g. thematic maps, open street map data, shape files, or any other data that provide georeferenced information. Typically, these data are generated automatically (e.g. with feature extraction from aerial images) but may require manual effort. It is important that the data are reliable and up to date to ensure a correct classification of ground points. The geographic registration between these thematic data and the 3D point cloud could include a certain degree of uncertainty and varying accuracies (e.g. derivations up to 2 m). To enhance the classification, multiple data sources can be used to get precise ground classification results. In contrast, building and vegetation detection cannot be performed reliably with additional geospatial data. Reasons are the 3D characteristics of vegetation and building points (e.g. horizontal overlapping), the missing data accuracy (e.g. rapid change of vegetation structures), and incompleteness of additional geospatial data.

The classification of ground points is performed in three steps. First, all available data sources are determined and prepared. For instance, Web Map Services (WMSs) are used to

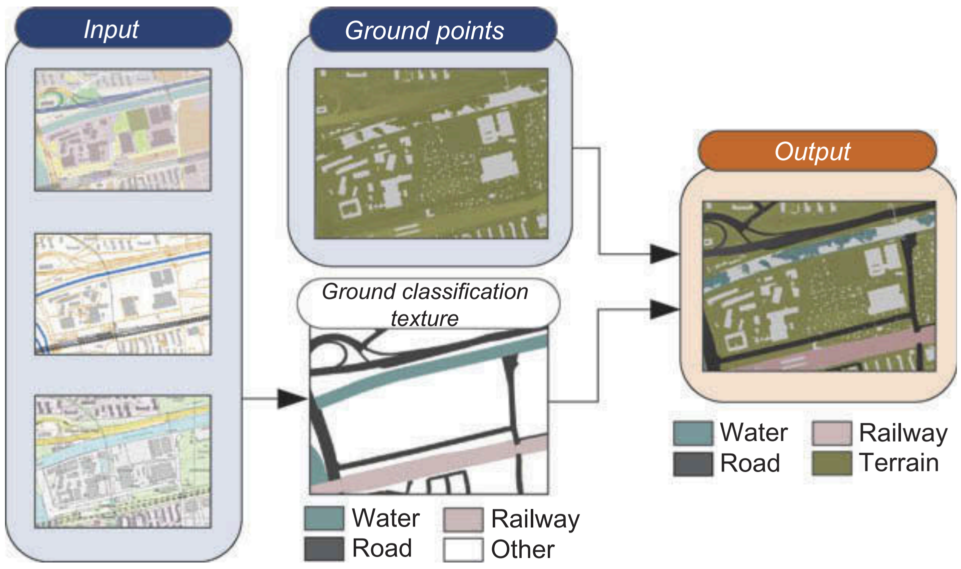


Figure 3. Ground classification workflow using multiple geospatial data sources to subdivide ground points into infrastructure, water, and terrain points.

obtain maps with thematic information. Priorities for each input source and object class are used to handle redundant, contradictory, and overlapping information. For example, road network information is available in thematic maps and open street map data, but differs in coverage and accuracy. Second, a *ground classification texture* is generated by combining object class information and priorities from the input data. In the last step, the ground classification texture is used to determine the object class information for each ground point. Figure 3 illustrates the ground classification process for terrain, road, railway, and water surfaces.

3.3. Building and vegetation classification

An iterative multi-pass approach is used to identify building and vegetation points. It performs the most challenging part of the object class segmentation pipeline and is illustrated in Algorithm 1. Traditional approaches use per-point or per-segment feature values for the classification. Typical features are normal distribution, regularity, horizontality, and flatness (Zhou and Neumann 2008) of segments or points. They are derived from the 3D point cloud topology of the local point proximity. In general, planar segments are classified as roof elements and scattered segments are classified as vegetation. This approach works well for large segments but becomes unreliable for small segments, i.e. with less than 50 points. These segments are difficult to analyse with current state-of-the-art approaches and therefore often labelled as unclassified points. Regions with an accumulation of unclassified points typically occur due to

- scattered roof segments (e.g. stacks, antennas, roof constructions),
- mixed roof and tree-covered areas (e.g. trees covering roofs, trees in the courtyard), and
- building facades with a small number of points (e.g. 1 point s/m^2).

To overcome this limitation, we propose an iterative multi-pass classification approach that benefits in each iteration pass from classified points in previous passes. Each pass analyses aspects of the 3D point cloud topology that are specific for individual object classes.

Algorithm 1: Building and vegetation classification.

```
Data: input segments ($S_{\text{input}}$); ground points ($P_{\text{g}}$);
large segment size threshold (minSize);
Result: building segments ($S_{\text{b}}$); vegetation segments ($S_{\text{v}}$);
forall the $s_i \in S_{\text{input}}$ do
    if #points of $s_{\text{i}}$ > minSize then
        $(S_{\text{b}}, S_{\text{v}}, S_{\text{posVeg}}) \xleftarrow{\text{add}}$ largeSegmentAnalysis ($s_{\text{i}}, P_{\text{g}}$);
    else
        $S_{\text{unclassified}} \xleftarrow{\text{add}} s_{\text{i}}$;
    end
end
// see Algorithm 2
$(S_{\text{v}}, S_{\text{unclassified}}) \leftarrow$ vegetationAnalysis ($S_{\text{v}}, S_{\text{posVeg}}, S_{\text{unclassified}}$);
forall the $s_i \in S_{\text{unclassified}}$ do
    $(S_{\text{b}}, S_{\text{v}}) \xleftarrow{\text{add}}$ smallSegmentAnalysis ($s_{\text{i}}, P_{\text{g}}, S_{\text{v}}, S_{\text{b}}$);
end
```

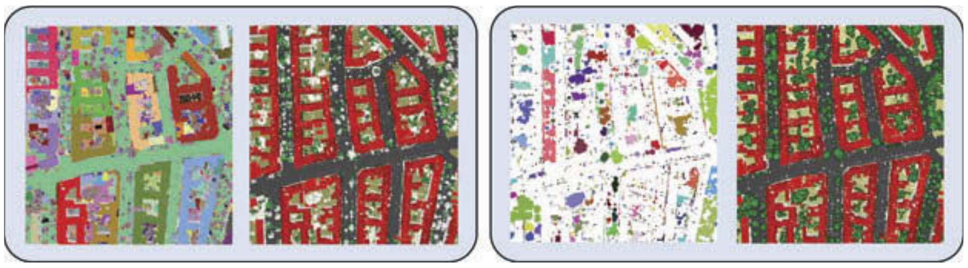
At first, only large segments that can be clearly or most likely assigned to an object class are identified in the *large segment analysis* pass. Second, a more precise *vegetation analysis* pass is performed using an additional segmentation and taking into account already classified vegetation and building points. In the third pass, a *small segment analysis* is performed that considers a larger proximity for the analysis.

3.3.1. *Large segment analysis*

All large segments (i.e. with more than 50 points) are analysed in relation to already detected ground points. For each point in a segment, all subjacent ground points are determined. If a segment contains many points above the ground, it is identified as vegetation. This assumption can be made because lidar rays partially run through vegetation up to the ground, in contrast to building structures where no subjacent ground points can be found. Consequently, segments containing many points without subjacent ground points can be most likely classified as building segments. Exceptions to this assumption are dense vegetation areas where lidar rays do not reach the ground surface and tree-covered areas that are captured with image-based remote-sensing technologies. For that reason, an additional validation pass is performed to analyse large segments without subjacent ground points. For each point, a flatness attribute is computed based on the normal variation to the nearest neighbour points. If the majority of points belong to a scattered region, the overall segment is classified as possible vegetation, otherwise as building. The results of the large segment analysis are building, vegetation, possible vegetation, and unclassified segments. Figure 4(*a*) illustrates the input segments (left) and detected building and vegetation segments (right).

3.3.2. *Vegetation analysis*

In contrast to areas with urban structures, tree-covered areas result in a large number of small segments (Figure 4(*a*) left). These segments are difficult to classify due to similarities to other small segments (e.g. from buildings, stacks, antennas, roof constructions and city furniture). Consequently, a segment-based analysis of vegetation points using segments



(*a*) (*b*)

Figure 4. 3D point cloud segments (left) and resulting object classes (right) for large segment (*a*) and vegetation (*b*) analysis.

Algorithm 2: Vegetation analysis.

```
Data: vegetation segments ($S_v$); possible vegetation segments ($S_{posVeg}$);
unclassified segments ($S_{unclassified}$); large segment size threshold (minSize);
Result: vegetation segments ($S_v$); unclassified segments ($S_{unclassified}$);
$P_{all} \leftarrow$ collectPoints ($S_v$, $S_{posVeg}$, $S_{unclassified}$);
$S_{new} \leftarrow$ nearestNeighbourSegmentation ($P_{all}$);
forall the $s_i \in S_{new}$ do
    if #points of $s_i >$ minSize then
        number $\leftarrow$ countUnclassifiedPoints ($s_i$);
        if number $<$ threshold then
            $S_v \xleftarrow{add} s_i$;
        else
            $S_{unclassified} \xleftarrow{add} s_i$;
        end
    else
        $S_{unclassified} \xleftarrow{add} s_i$;
    end
end
```

of the previous pass becomes unreliable. Therefore, an additional segmentation pass is performed taking into account typical 3D point cloud topology for vegetation areas and already detected vegetation segments. The segmentation is performed on already detected vegetation, possible vegetation, and unclassified segments without smoothness parameters. As a result, segment clusters are merged and small segments are added to already detected vegetation segments (Figure 4(*b*) left). In the next step, all segments that exceed a defined size (e.g. 50 points) are analysed because these segments allow a reliable classification. This analysis is performed with respect to the amount of vegetation and possible vegetation points that were detected in the large segment analysis. If the majority of points belong to these object classes, all points in the segment are assigned to the object class vegetation. Algorithm 2 illustrates the workflow and Figure 4(*b*) shows segmentation and vegetation analysis results. The remaining unclassified segments are very small and not connected to already detected vegetation areas.

3.3.3. Small segment analysis

In this pass, all remaining unclassified segments are analysed taking into account already detected building, vegetation and ground points. These segments typically tend to occur at small building roof elements, facades, low vegetation, and city furniture (Figure 4(*b*) – right). Small roof elements can be detected if all neighbouring points are already detected building points. Facade segments are more difficult to detect, due to an incomplete surface and a small number of facades points in general. The detection is performed by specifying a point neighbourhood volume, similar to a tube, with a small radius (e.g. 0.5 m). A point is classified as facade point if building points can be found above and building or ground points can be found below a point. If the majority of points in a segment fulfil this property, all segment points are identified as facade points belonging to the object class building. All remaining segments belong to low vegetation or city furniture. Assumptions are made by considering already classified ground points. For instance, cars can be found above or next to infrastructure points.

4. Out-of-core and GPU-based processing

The algorithms and processes of the object class segmentation have different memory and computation time requirements. This becomes particularly relevant when the data amount increases, i.e. billions of points need to be processed. To perform the classification for massive 3D point clouds on standard consumer hardware, out-of-core processing strategies and a parallel GPU-based processing of time-consuming processes have been implemented.

To overcome main memory limitations, an out-of-core quadtree is used to handle the entire 3D point cloud and enable fast data integration, subdivision, and updates. Each quadtree node has an axis-aligned bounding box and is either an inner node with up to four child nodes, or a leaf node with a subset of the points of the parent node. The bounding box of the root node is derived from the spatial distribution of the data. The subdivision of nodes is adaptive and depends on the number of points in a spatial area. To construct the out-of-core quadtree, input data is divided into subsets that can be processed in main memory and serialized to secondary storage (e.g. 100 M points). A sequential processing of all subsets allows building up the entire tree structure on secondary storage. All points and corresponding attributes are stored uncompressed in leaf nodes to enable fast access for further processing. In general, the size and capacity of the out-of-core quadtree are only limited by the available secondary storage capacity (e.g. hard disk). Figure 5 illustrates the structure of a quadtree that stores points in its leaf nodes. The depth of the quadtree can be predefined (e.g. 8 levels) or adaptive depending on the spatial distribution of points (e.g. 1024 points per node). The quadtree enables an adaptive selection of data tiles that fit into available main memory and enable an in-memory processing. To ensure a correct classification for points close to tile boundaries, each tile includes overlapping areas of neighbouring tiles. Based on the distribution of the 3D point cloud, suitable tiles that fit into available main memory are selected and processed in a sequential order. The bottlenecks of the object class segmentation are multiple local neighbourhood requests for a large number of points. Multiple requests are necessary, due to varying parameters:

- request volume (e.g. sphere, tube) and direction (e.g. below or above requested points);
- volume size (e.g. 10 m, 0.5 m); and
- requested object class (e.g. ground, vegetation, building).

The GPU facilitates massive parallel processing using multiple processors and dedicated memory directly on the device. For it, we use NVIDIA’s (http://www.nvidia.com) Compute Unified Device Architecture (CUDA) as parallel computing architecture that provides a

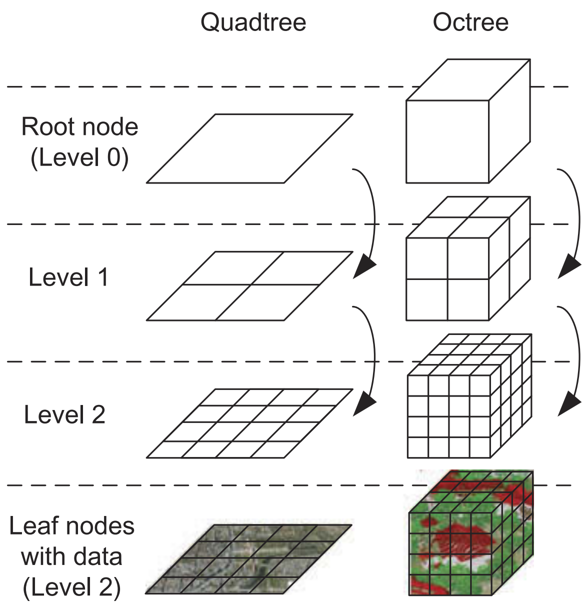


Figure 5. Illustration of the quadtree and octree structures that are used to arrange the 3D point clouds.

flexible programming API for general purpose computations on GPUs. CUDA enables the parallelization of algorithms by dividing the input data into chunks that are processed in parallel by a collection of threads (Farber 2011). Other parallel GPU computing models and APIs such as OpenCL or compute shader could also be used.

We use an octree as spatial data structure to improve the performance of local neighbourhood requests. Octrees have a similar concept to quadtrees and divide the data according to three dimensions (Figure 5). Each node has up to eight child nodes that can be traversed to enable fast access to subareas of the data. The process is divided into five steps. At first, all points belonging to object classes that should be processed are transferred to the GPU memory and arranged in an octree. Second, request parameters for volume, size, and object class are defined. Third, all points whose local neighbourhood should be analysed are transferred to GPU memory. Fourth, these points are assigned to parallel executed threads that traverse the octree structure and determine all points that fulfil the specified request parameter. Finally, results are stored in a data structure on the GPU and transferred back to CPU memory. Special cases occur, if the number of points in the requested area exceeds the available GPU memory capacity. To overcome this limitation, a subdivision into suitable chunks is necessary. The presented approach performs best for requests with a large number of query points (e.g. 100 k points) to use the full processing capacities of the GPU. We implemented our algorithm also for the CPU. Our measurements have shown that the GPU implementation performs on average 25 times faster.

5. Results and evaluation

In this section, we present classification results using an out-of-core rendering technique for massive 3D point clouds. In addition, we evaluate the accuracy and performance of the object class segmentation. We use a 3D point cloud from an aerial lidar scan with a point density of 5–10 points/m^2. A detailed evaluation is performed for three tiles (480,000 m^{-2}) representing different environment characteristics, i.e. a tree-covered, suburban, and downtown area. Results are listed in Table 2 and include the point

Table 2. Object class and segment statistics for a tree-covered (Tile 1), suburban (Tile 2), and downtown (Tile 3) area with different characteristics (Figure 7) and the overall 3D point cloud of Berlin (Figure 6).

	Tile 1		Tile 2		Tile 3		All tiles	
Number points	5354 K	(100%)	2819 K	(100%)	2414 K	(100%)	4601 M	(100%)
Terrain points	2464 K	(46.0%)	1831 K	(65.0%)	571 K	(23.2%)	2399 M	(52.1%)
Infrastructure points	182 K	(3.4%)	197 K	(7.0%)	652 K	(26.5%)	428 M	(9.3%)
Water points	25 K	(0.5%)	12 K	(0.4%)	8 K	(0.3%)	55 M	(1.2%)
Vegetation points	2587 K	(48.3%)	456 K	(16.2%)	160 K	(6.5%)	1228 M	(26.7%)
Building points	96 K	(1.8%)	323 K	(11.5%)	1023 K	(41.5%)	491 M	(10.7%)
Number Segments	1154 K		270 K		106 K		975 M	
Large Segments	18 K		5 K		3 K		11 M	
Small segments	1136 K		265 K		103 K		964 M	

distribution for each object class and the number of segments that result from the segmentation pass. The point density varies because of multiple lidar returns, especially caused by vegetation.

5.1. *Visualization*

An interactive rendering is essential for visualizing, assessing, and interpreting massive 3D point clouds (Kreylos, Bawden, and Kellogg 2008). We extended the rendering technique of Richter and Döllner (2010) and Goswami et al. (2010) to individually customize the stylization and visualization for different object classes.

All points are organized in a hierarchical data structure with multiple levels-of-detail (LODs) of the 3D point cloud, called multi-resolution tree structure. This structure has only minor similarities to the data structure that is used for the object class segmentation. The multi-resolution tree structure enables selection of visible and for rendering relevant LODs. In addition, the selection of different object classes is possible. The multi-resolution tree structure arranges points with corresponding attributes (e.g. colour, normal and classification information) in buffer objects that are associated with LOD representations. These buffer objects are transferred to the GPU memory and provide all necessary information for an interactive stylization of object classes. To overcome main memory limitations, a swapping of LOD representations between main memory and secondary storage is performed by separate threads. This is essential to keep only for the rendering relevant 3D point clouds in main memory.

Figure 6 shows the results of the object class specific visualization. Object classes are illustrated with individual colour schemes: buildings, red; vegetation, green; terrain, yellow; water, blue; and infrastructure grey. Building points are represented by splats orientated towards the normal vector of the point. Vegetation points are rendered with a silhouette rendering technique for 3D point clouds (Xu et al. 2004). All other object classes are illustrated with horizontally orientated splats. Figure 6(*a*) shows the input 3D point cloud coloured with aerial images while (*b*), (*c*), and (*d*) show the classification result for the urban area of Berlin.

5.2. *Accuracy evaluation*

To evaluate the accuracy of object class segmentation results, we used manually classified 3D point clouds as ground truth. We implemented a tool that allows a human operator to

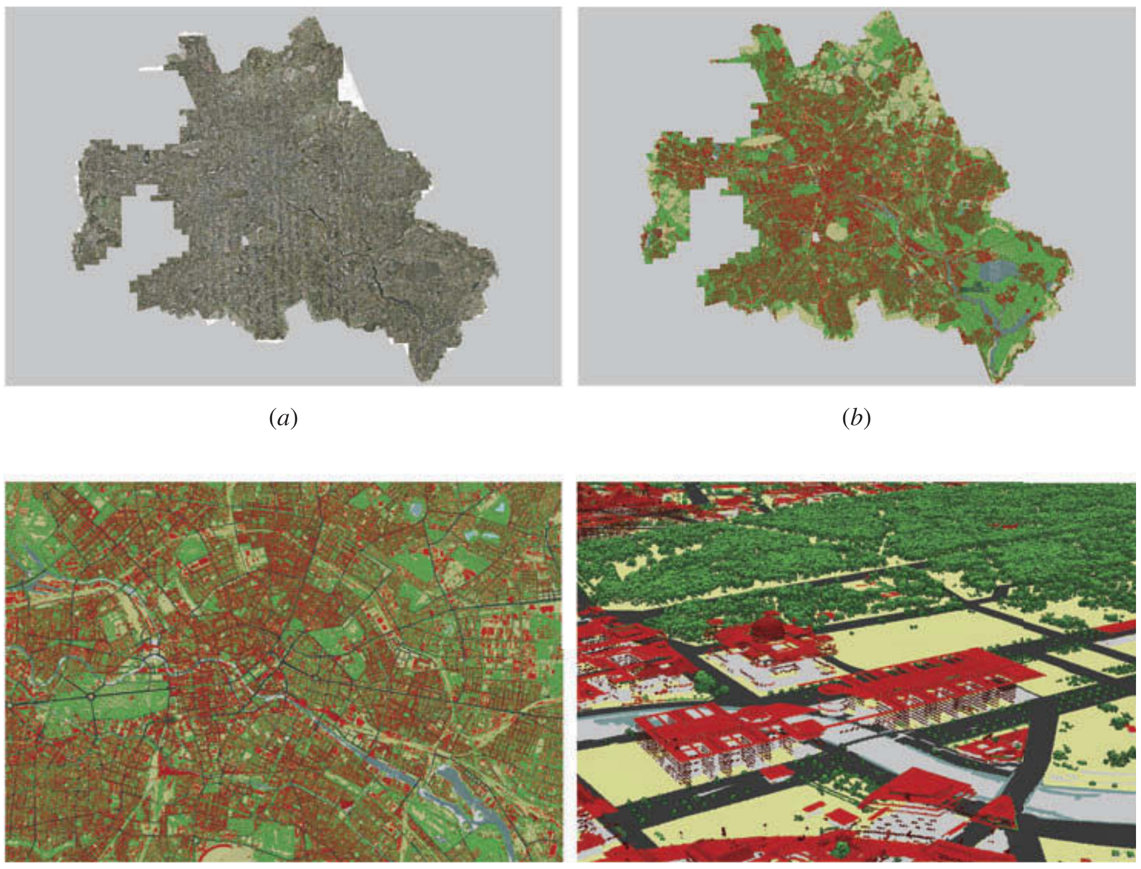


Figure 6. Illustration of the object class segmentation results for the urban area of Berlin with almost 5 billion points. (*a*) 3D point cloud with colours from aerial images. (*b*) 3D point cloud coloured with object class information. (*c*) Detailed view of upper right. (*d*) Detailed view of lower left.

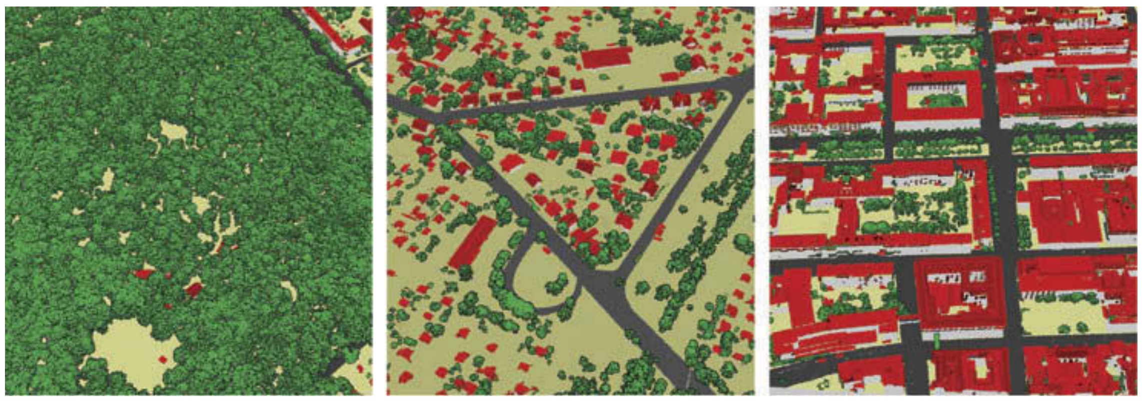


Figure 7. Illustration of three tiles with different characteristics: Tile 1, tree-covered (*a*); Tile 2, suburban (*b*); and Tile 3, downtown area (*c*).

classify points into the object classes ground, vegetation, and building. A manual classification of ground points into terrain, infrastructure, and water points was not performed. The classification and evaluation was done for a subarea (25%) of the three tiles (Figure 7) to limit the manual classification effort.

Table 3. Object class segmentation evaluation based on manually classified 3D point clouds used as ground truth (GT) for tree-covered, suburban, and downtown areas. Precision, recall, and total error are derived from true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) for vegetation, building, and ground points.

	#GT	#Result	TP	FP	FN	TN	Precision	Recall	Total error
Tile 1									
Vegetation	556,697	557,301	555,621	1680	1076	587,634	99.70%	99.81%	0.24%
Building	31,657	30,721	30,511	210	1146	1,114,144	99.32%	96.38%	0.12%
Ground	557,657	557,989	556,954	1035	703	587,319	99.81%	99.87%	0.15%
Tile 2									
Vegetation	88,202	89,298	83,414	5884	4788	432,138	93.41%	94.57%	2.03%
Building	98,515	99,230	94,517	4713	3998	422,996	95.25%	95.94%	1.66%
Ground	339,507	337,696	335,209	2487	4298	184,230	99.26%	98.73%	1.29%
Tile 3									
Vegetation	98,412	93,204	90,874	2330	7538	448,969	97.50%	92.34%	1.80%
Building	159,075	160,449	157,093	3356	1982	387,280	97.91%	98.75%	0.97%
Ground	292,224	297,077	291,045	6032	1179	251,455	97.97%	99.60%	1.31%

Precision, recall, and error rates for each object class are listed in Table 3. The precision for detection of vegetation and building points indicates that the presented approach performs best for tree-covered and downtown areas where vegetation and building structures are mainly separated. However, also suburban areas with building and vegetation structures with similar sizes and overlapping volumes can be separated with a precision of 93% for vegetation and 95% for buildings. The ground detection performs best for the vegetation and suburban areas because the number of ground points, in relation to the total number of input points, is much higher in contrast to downtown areas. Ground segments are more connected and in general larger, which is advantageous for the ground detection stage. The results show that the presented object class segmentation provides a robust classification with a maximum total error of 2.03% for vegetation, 1.66% for building, and 1.31% for ground point detection.

We identified some environment scenarios, where the object class segmentation does not work reliably. For example, large vegetation covered balconies are detected as vegetation, heavy smoke (e.g. from power plants) is classified as vegetation, and roofs with large window areas are not classified correctly. Also, low point densities (<5 points/m^2) could cause incorrect results for small structures. Figure 6(*d*) shows that water surfaces have varying point densities because water reflects lidar rays poorly.

5.3. Performance evaluation

This section presents the performance evaluation of the object class segmentation. First, performance measurements for the three characteristic tiles that are processed in main memory are presented and discussed. Second, the performance of the overall workflow to classify a 3D point cloud of the urban area of Berlin with almost 5 billion points is evaluated. All tests were performed on an Intel Xeon CPU with 2.66 GHz and 12 GB main memory. For GPU computations a NVIDIA GeForce GTX 480 with 1536 MB device memory and 480 CUDA cores is used.

The measurements in Table 4 show that the preprocessing and segmentation time are only slightly affected by the point cloud characteristics. Also the performance of the ground

Table 4. Object class segmentation performance for three tiles with different characteristics (Figure 7) and the overall 3D point cloud of Berlin (Figure 6) in seconds (all points per million points).

Task	Tile 1	Tile 2	Tile 3	All tiles
Point cloud import	1.39 s/0.26 s	0.54 s/0.19 s	0.60 s/0.25 s	22.38 m/0.29 s
Preprocessing	5.41 s/1.15 s	2.31 s/1.04 s	2.33 s/1.16 s	87.12 m/1.14 s
Segmentation	35.69 s/7.61 s	13.79 s/6.20 s	11.52 s/5.73 s	534.83 m/6.97 s
Ground detection	1.90 s/0.40 s	0.90 s/0.41 s	0.51 s/0.25 s	30.78 m/0.40 s
Ground classification	1.59 s/0.34 s	0.76 s/0.34 s	0.74 s/0.37 s	26.33 m/0.34 s
Large segment analysis	10.71 s/2.28 s	2.86 s/1.28 s	3.16 s/1.57 s	125.63 m/1.64 s
Vegetation analysis	5.21 s/1.11 s	0.60 s/0.27 s	0.35 s/0.17 s	46.84 m/0.61 s
Small segment analysis	34.85 s/7.43 s	11.84 s/5.32 s	11.25 s/5.60 s	475.42 m/6.20 s
Point cloud export	0.71 s/0.13 s	0.37 s/0.13 s	0.31 s/0.13 s	12.32 m/0.16 s
Overall classification	**97.46 s/20.71 s**	**33.97 s/15.18 s**	**30.77 s/15.23 s**	**1361.65 m/17.75 s**

detection pass is only affected by the overall number of points. The increased processing time for the vegetation analysis for Tile 1 is caused by the large number of vegetation segments. The different processing times for the small segment analysis result from the total number of small segments, e.g. facades, that need to be processed. In summary, the object class segmentation of urban areas performs best due to the lower number of segments.

The overall object class segmentation workflow for the 3D point cloud of Berlin takes less than 23 hours. Processing without GPU accelerated algorithms would require about 250 hours. The average throughput of our system is 3.3 million points per minute. However, there is still potential to increase the processing performance, e.g. by performing the 3D point cloud segmentation pass on the GPU.

6. Conclusions and future work

We have presented a concept and implementation for object class segmentation of massive 3D point clouds for urban areas. It is based on a processing pipeline that classifies the data into building, vegetation, terrain, water, and infrastructure points. In particular, the detection of building and vegetation points is performed with an iterative multi-pass algorithm taking into account 3D point cloud topology. The number of unclassified points is reduced due to the iterative process. The key feature of our approach is that 3D point clouds are robustly processed without additional per-point attributes or training data sets. Thematic maps are used to split detected ground points into disjoint object classes. Our accuracy evaluation indicates that the approach works reliably for areas with different characteristics such as tree-covered, suburban, and downtown areas. We have shown that out-of-core concepts and GPU-based processing schemes enable the object class segmentation, even for massive 3D point clouds. Object class segmentation facilitates analysis and provides a fundamental instrument to improve applications, systems, and workflows. One application field that benefits from 3D point clouds with object class information is disaster management. Here, the fast analysis of large 3D point clouds is essential to estimate object class specific damage to buildings, infrastructure, or forests. Another application is the selective update for geospatial data, such as surface models, maps, and 3D city models.

We have also presented a rendering system to enhance the exploration and visualization of massive 3D point clouds using object class information. Future research can focus on performing the entire object class segmentation on the GPU to increase the performance. Furthermore, the analysis of 3D point clouds from different points in time would enable us to determine dynamic and static objects in urban environments. In addition, a fusion of 3D point clouds captured with mobile and aerial remote-sensing systems generates more dense 3D point clouds that allow for a more detailed object class segmentation (e.g. for city furniture, cars, and facades). This requires additional analysis passes and results in more expensive analysis and processing. However, it allows us to generate more detailed and realistic 3D models (e.g. 3D city models) that can be used in a variety of application domains.

Funding

This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, within the InnoProfile Transfer research group '4DnD-Vis'. We would like to thank virtualcitySYSTEMS for providing lidar data.

References

Alexander, Cici, Kevin Tansey, Jörg Kaduk, David Holland, and Nicholas J. Tate. 2011. "An Approach to Classification of Airborne Laser Scanning Point Cloud Data in an Urban Environment." *International Journal of Remote Sensing* 32: 9151–9169.

Carlberg, Matthew, Peiran Gao, George Chen, and Avideh Zakhor. 2009. "Classifying Urban Landscape in Aerial Lidar Using 3D Shape Analysis." In *16th IEEE International Conference on Image Processing (ICIP)*, 1681–1684, Cairo. New York, NY: IEEE.

Charaniya, Amin P., Roberto Manduchi, and Suresh K. Lodha. 2004. "Supervised Parametric Classification of Aerial LiDAR Data." In *Computer Vision and Pattern Recognition Workshop*, 25–32, Washington, DC. New York, NY: IEEE.

Clode, Simon, and Franz Rottensteiner. 2005. "Classification of Trees and Powerlines from Medium Resolution Airborne Laserscanner Data in Urban Environments." In *APRS Workshop on Digital Image Computing (WDIC)*, edited by B. C. Lovell and A. J. Maeder, 97–102. Brisbane: The University of Queensland.

da Silva, Claudionor R., Jorge A. S. Centeno, and Maria J. Henriques. 2011. "Automatic Road Extraction on Aerial Photo and Laser Scanner Data." *International Conference on Environmental and Computer Science* 19: 99–103.

Douillard, Bertrand, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. 2011. "On the Segmentation of 3D LiDAR Point Clouds." In *International Conference on Robotics and Automation*, 2798–2805, Shanghai. New York, NY: IEEE.

Farber, Rob. 2011. *CUDA Application Design and Development*. Burlington, MA: Morgan Kaufmann.

Golovinskiy, Aleksey, G. Vladimir Kim, and Thomas Funkhouser. 2009. "Shape-based Recognition of 3D Point Clouds in Urban Environments." In *12th International Conference on Computer Vision*, 2154–2161, Kyoto. New York, NY: IEEE.

Goswami, Prashant, Yanci Zhang, Renato Pajarola, and Enrico Gobbetti. 2010. "High Quality Interactive Rendering of Massive Point Models Using Multi-way kd-Trees." In *18th Pacific Conference on Computer Graphics and Applications*, 93–100, Hangzhou. New York, NY: IEEE.

Hoppe, Hugues, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1992. "Surface Reconstruction from Unorganized Points." *ACM SIGGRAPH Computer Graphics* 26: 71–78.

Jiang, Jingjue, Zuxun Zhang, and Ying Ming. 2008. "Filtering of Airborne Lidar Point Clouds for Complex Cityscapes." *Geo-spatial Information Science* 11: 21–25.

Kreylos, Oliver, Gerald W. Bawden, and Louise H. Kellogg. 2008. "Immersive Visualization and Analysis of LiDAR Data." In *4th International Symposium on Advances in Visual Computing*, edited by George Bebis, Richard D. Boyle, Bahram Parvin, Darko Koracin, Paolo Remagnino, Fatih Murat Porikli, Jörg Peters, James T. Klosowski, Laura L. Arns, Yu Ka Chun, Theresa-Marie Rhyne, and Laura Monroe, 846–855, Las Vegas, NV. Berlin: Springer.

Leberl, Franz, Arnold Irschara, Thomas Pock, Philipp Meixner, Michael Gruber, Susanne Scholl, and Alexander Wiechert. 2010. “Point Clouds: Lidar versus 3D Vision.” *Photogrammetric Engineering and Remote Sensing* 76: 1123–1134.

Lodha, Suresh K., Darren M. Fitzpatrick, and David P. Helmbold. 2007. “Aerial Lidar Data Classification using AdaBoost.” In *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 435–442, Montreal. New York, NY: IEEE.

Meixner, Philipp, Franz Leberl, and Mathieu Brédif. 2011. “Planar Roof Surface Segmentation Using 3D Vision.” In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, edited by Isabel F. Cruz, Divyakant Agrawal, Christian S. Jensen, Eyal Ofek, and Egemen Tanin, 9–15, Chicago, IL. New York, NY: ACM.

Meng, Xuelian, Nate Currit, and Kaiguang Zhao. 2010. “Ground Filtering Algorithms for Airborne LiDAR Data: A Review of Critical Issues.” *Remote Sensing* 2: 833–860.

Rabbani, Tahir, Frank van den Heuvel, and George Vosselman. 2006. “Segmentation of Point Clouds Using Smoothness Constraint.” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 36: 248–253.

Raber, George T., John R. Jensen, Steven R. Schill, and Karen Schuckman. 2002. “Creation of Digital Terrain Models Using an Adaptive Lidar Vegetation Point Removal Process.” *Photogrammetric Engineering & Remote Sensing* 68: 1307–1315.

Richter, Rico, Jan E. Kyprianidis, and Jürgen Döllner. 2012. “Out-of-Core GPU-based Change Detection in Massive 3D Point Clouds.” *Transactions in GIS*. http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9671.2012.01362.x/abstract

Richter, Rico, and Jürgen Döllner. 2010. “Out-of-core Real-time Visualization of Massive 3D Point Clouds.” In *7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, edited by James E. Gain and Patrick Marais, 121–128, Franschoek. New York, NY: ACM.

Samadzadegan, Farhad, Behnaz Bigdeli, and Pouria Ramzi. 2010. “A Multiple Classifier System for Classification of LiDAR Remote Sensing Data Using Multi-class SVM.” In *Multiple Classifier Systems*, edited by Neamat El Gayar, Josef Kittler, and Fabio Roli, 254–263, Cairo. Berlin: Springer.

Shao, Yi, and Liang Chen. 2008. “Automated Searching of Ground Points from Airborne Lidar Data Using a Climbing and Sliding Method.” *Photogrammetric Engineering and Remote Sensing* 74: 625–635.

Shapovalov, Roman, and Alexander Velizhev. 2011. “Cutting-Plane Training of Non-associative Markov Network for 3D Point Cloud Segmentation.” In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, edited by Michael Goesele, Yasuyuki Matsushita, Ryusuke Sagawa, and Ruigang Yang, 1–8, Hangzhou. New York, NY: IEEE.

Shapovalov, Roman, Alexander Velizhev, and Olga Barinova. 2010. “Non-Associative Markov Networks for 3D Point Cloud Classification.” *Photogrammetric Computer Vision and Image Analysis* 38 (3A): 103–108.

Sithole, George, and George Vosselman. 2004. “Experimental Comparison of Filter Algorithms for Bare-Earth Extraction from Airborne Laser Scanning Point Clouds.” *ISPRS Journal of Photogrammetry and Remote Sensing* 59: 85–101.

Sithole, George, and George Vosselman. 2005. “Filtering of Airborne Laser Scanner Data Based on Segmented Point Clouds.” In *Laser Scanning 2005*, edited by G. Vosselman and C. Brenner, 66–71, Enschede, September 12–14.

Snavely, Noah, Rahul Garg, Steven M. Seitz, and Richard Szeliski. 2008. “Finding Paths Through the World’s Photos.” *ACM Transactions on Graphics* 27: 11–21.

Wehr, Aloysius, and Uwe Lohr. 1999. “Airborne Laser Scanning – an Introduction and Overview.” *ISPRS Journal of Photogrammetry and Remote Sensing* 54: 68–82.

Xu, Hui, Minh X. Nguyen, Xiaoru Yuan, and Baoquan Chen. 2004. “Interactive Silhouette Rendering for Point-Based Models.” In *Eurographics Symposium on Point-Based Graphics*, edited by M. Alexa and S. Rusinkiewicz, 13–18, Zurich. Geneva: Eurographics.

Zhou, Qian-Yi, and Ulrich Neumann. 2008. “Fast and Extensible Building Modeling from Airborne Lidar Data.” In *16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, edited by Walid G. Aref, Mohamed F. Mokbel, and Markus Schneider, 1–8, Irvine, CA. New York, NY: ACM.